APPLICATION FOR UNITED STATES PATENT

in the name of

Bridget J. Frey
Michael Wei-Chin Young

of

Plumtree Software, Inc.

for

DELEGATED AUTHENTICATION USING A GENERIC APPLICATION-LAYER NETWORK PROTOCOL

Law Office of Richard A. Dunning, Jr.
323M Sharon Park Drive, Suite 208
Menlo Park, CA 94023
Tel.: 630.234.1034
Fax: 630.234.8427

ATTORNEY DOCKET:
PLM007001

DATE OF DEPOSIT: _____

EXPRESS MAIL NO.: ET 161000191 US

DELEGATED AUTHENTICATION USING A GENERIC APPLICATION-LAYER NETWORK PROTOCOL

BACKGROUND

This disclosure relates generally to information retrieval and distribution systems. More specifically, it relates to a method and system for assembling and distributing content components generated by multiple resource servers having diverse authentication requirements.

It is common for today's enterprise networks to comprise scattered arrangements of different hardware and software systems. This is due to the ever-changing data management needs of corporate enterprises, and to continuing advances in the computing hardware and software available to meet those needs. Commonly, different entities within an enterprise (for example, different departments or work sites) have disparate software applications, groupware systems, or data maintenance architectures/procedures, such that information created or maintained by one entity is not usable by another entity.

Corporate portals, also referred to as intranet portals, have been introduced to increase the accessibility and usability of information stored across the heterogeneous systems of an enterprise network. A corporate portal, which is usually overlaid onto an existing enterprise network, is designed to extract content from disparate systems on the enterprise network and to allow easier, personalized access to that content by end users. It is to be appreciated that while the features and advantages of the implementations described herein are particularly advantageous for corporate portal environments, enhancing their speed, openness, scalability, and stability, the features and advantages of the implementations are also applicable in other environments, such as with personalized "Web portals" that serve broad user bases. By way of example and not by way of limitation, one example of a corporate portal is the Plumtree Corporate Portal available from Plumtree Software, Inc. of San Francisco, California, while examples of personalized Web portals are typified by the MyYahoo! resource from Yahoo, Inc. of Sunnyvale, California and MyExcite from At Home Corp. of Redwood City, California. Corporate portals are also described in commonly assigned U.S. Ser. No. 09/896,039, filed June 29, 2001, which is incorporated by reference herein.

FIG. 1 shows a simplified view of an exemplary user screen 102 associated with a corporate portal system, comprising a plurality of content components 104-110. A content component refers to any content that is assembled, along with other content components, into a unified body of content. In the example of FIG. 1, a company news content component 104 includes an HTML display of news that is extracted, for example, from one or more company news servers, and arranged for display to the end user. A company stock quote content component 106 comprises an HTML display of a stock quote for the company and its competition that is extracted, for example, from a stock quote server. Also shown in FIG. 1 is an email content component 108 and a customer relationship management (CRM) content component 110. According to the end user's ID 112, the corporate portal displays the content components 104-110 in a personalized arrangement (for example, news at the upper left, company stock quote in the upper right, and so on) and also selects the information within each content component based on the user's ID (for example, showing the user's personal e-mail account only, showing sports news on top of world news, showing only the user's personal CRM information, and so on). The user screen 102 of FIG. 1 would typically appear after the user (Jane Smith) has logged into the corporate portal system by supplying a user name and password.

More generally, the content components themselves can be any information communicable via any generic application-layer network protocol such as Hypertext Transfer Protocol (HTTP), Secure Hypertext Transfer Protocol (HTTPS), File Transfer Protocol (FTP), Wireless Application Protocol (WAP), and the like. Information communicable via a network includes text information, image information, Extensible Markup Language (XML), Hypertext Markup Language (HTML), or any other type of information that can be stored in a computer file, including images, sounds, and video. Throughout this specification we refer to any information sent over a network as content. We use the term content component to refer to any content that is assembled, along with other content components, into a unified body of content.

An exemplary content component is the HTML output generated by a script that communicates with an email client application. An email client application sends and receives email. Such applications usually let users compose email, and store email addresses in an address book. This script provides an HTML interface to the email client application.

This script is hosted by the computer hosting the email application. This script generates HTML displaying the user's email messages, along with HTML allowing the user to compose and send email messages. This script can communicate with the email application through the application's programming interface. In this example, the HTML generated by the script is the content component (see, for example, FIG. 1, content component 108).

Other exemplary content components are two types of HTML generated by a program that communicates with a database application. This program can be hosted by the same computer hosting the database application. The database application stores and maintains a database of information organized into records. This program can communicate with the database application via the application's interface. This program generates HTML that allows the user to search for database records. For this case, the content component is a query box. This program also generates HTML that displays database records to the user. For this case, the content component is a view of the database records (see, for example, FIG. 1, content component 110). Further examples of content components include, but are not limited to, resources generated by a calendar application, a workflow application, a database storing proprietary personal information, a database storing proprietary business information, a database storing secure personal information, a database storing secure business information, an e-business application, and the like.

Content components are obtained from servers referred to herein as "resource servers." In some cases, resource servers may be secure, so that security credentials are required to gain access to the content on a secure resource server. FIG. 2 shows a system 200 for delivering personalized content according to a prior art method. A plurality of secure resource servers 202-208, each protected by an authenticator 222-228, host various types of content. Each authenticator 222-228 authenticates each user before allowing access to the protected server. Referring to FIG. 2, a CRM server 202 protected by an authenticator 222 hosts content such as customer lists and customer contact information. An email server 204 protected by an authenticator 224 hosts content such as email messages for a group of users. A stock quotes server 206 protected by an authenticator 226 hosts content such as stock quotes and charts. A news server 208 protected by an authenticator 228 hosts content such as headlines and news features. A main process 216 within a Web server 210 maintains a list of

the types of content available from resource servers 202-408, and advertises these types of content to users.

Users employ user terminals 218A and 218B through 218N to access Web server 210 over a network 220 such as the Internet. A user establishes personalized settings in part by selecting certain of the types of content that are advertised by Web server 210. Subsequent to this personalization step, the user sends a request for personalized content to Web server 210. The personalized content can include content residing upon secure resource servers. Therefore, Web server 210 must provide security credentials to each secure resource server.

According to one prior art method Web server 210 simply forces the user to supply a security credential every time a secure resource server requires one. Since remembering multiple passwords or authentication methods is difficult for many users, users often write down or forget passwords, or use the same password for all of the secure resource servers. These user reactions create potential security and management problems.

According to another prior art method Web server 210 collects and stores the user security credentials for all of the secure resource servers. Whenever a secure resource server requires a security credential, Web server 210 simply provides all of the user's security credentials to the secure resource server. While relieving the user from entering his security credentials again and again, this technique creates significant security risks. For example, one of the secure resource servers could access another of the secure resource servers by spoofing the user.

To overcome these deficiencies in the prior art, techniques referred to as "Single-Sign-On" (SSO) have recently been developed. SSO techniques allow a user to access computers and systems to which he has permission through a single action, without the need to enter multiple passwords. One such technique is Kerberos, which allows a user to delegate authentication functions to another entity, such as Web server 210. One significant disadvantage of kerberos is that it does not support generic application-layer protocols, such as hypertext transfer protocol (HTTP) or file transfer protocol (FTP). Therefore a user cannot use Kerberos through a standard Web browser, such as Microsoft Internet Explorer or Netscape Navigator, without significant modification to the browser.

Some commercially-available off-the-shelf SSO products, such as Netegrity Siteminder and Securant Cleartrust, employ HTTP as the application-layer protocol, and so

are compatible with unmodified browsers. When a user first visits a web site employing this type of SSO product, the web site authenticates the user and then gives the user browser a token, such as a session cookie, that allows the user to access any other web site that is guarded by the same SSO product without going through the original web site or authenticating again. One disadvantage of such SSO products is that they work only with other SSO products. A token from one SSO product will not provide access to a secure web site that does not employ that SSO product.

Another prior art solution is Microsoft Passport, which provides a single authenticator, controlled by Microsoft. According to Passport, the secure resource server shares a secret key with the Microsoft Passport Authenticator. One disadvantage of Passport is that, in order to share a private key with the Microsoft Passport Authenticator, each secure resource server must enter a business relationship with Microsoft.

Another prior art solution is represented by the capability of some browsers to store a security credential required by a Web site, and to forward the security credential to the Web site automatically when the user directs the browser to that Web site. Such security credentials can include session cookies, persistent cookies, and digital certificates. One disadvantage of this approach is that it is not portable. In order to use this approach on a second computer, the user must install the credentials on that computer as well. In addition, if the second computer is available to other users, the credentials must be removed when the session ends to prevent a security breach.

SUMMARY

In general, in one aspect, a method, apparatus, and computer-readable media include receiving a signal representing a request from a remote user for a secure resource residing on a network employing a generic application-layer network protocol; determining, without the intervention of the user, the type of security credential required to access the secure resource; and sending a signal representing a second request to the secure resource, the second request including a security credential for the user of the type required to access the secure resource.

Particular implementations can include one or more of the following features. Particular implementations include authenticating the user before sending the signal representing the second request. Particular implementations include receiving a signal

representing a response to the second request; and sending a signal representing a result to the remote user, the result based on the response to the second request. The request includes a logon credential for the remote user, and particular implementations include authenticating the remote user based on the logon credential before sending the second request. The request includes a logon credential for the remote user and the type of security credential required to access the secure resource includes the logon credential, and particular implementations include sending the signal representing the second request to the secure resource, the second request including the logon credential. The request includes a logon credential for the remote user, and particular implementations include receiving a signal representing a single-sign-on (SSO) credential generated by a SSO provider based on the logon credential; and sending a signal representing the SSO credential to the secure resource when the type of credential required to access the secure resource includes the SSO credential. Particular implementations include sending a signal representing the SSO credential to the secure resource when the type of credential required to access the secure resource includes a second SSO token corresponding to a second SSO provider having a trust relationship with a first SSO provider corresponding to the SSO token. Particular implementations include receiving a signal representing a second SSO credential generated by a second SSO provider based on the first SSO credential; and sending a signal representing the second SSO credential to the secure resource when the type of credential required to access the secure resource includes the second SSO credential. The generic application-layer network protocol is hypertext transfer protocol. Particular implementations include receiving a signal representing data in response to the second request; and sending a signal representing at least a portion of the data to the remote user. The Web resource includes a Web site, and the data is hypertext mark-up language. Receiving includes receiving a signal representing a request from the remote user for a second secure resource residing on the network, and particular implementations include determining, without the intervention of the user, the type of security credential required to access the second secure resource; and sending a signal representing a third request to the second secure resource, the third request including a security credential for the user of the type required to access the second secure resource; and wherein the signals representing the second and third requests are sent concurrently. The types of security credentials included in the second and third requests differ. The types of security credentials included in the second

and third requests are the same. Particular implementations include receiving a signal representing the security credential from the user before receiving the signal representing the request. Particular implementations include storing the security credential at least until sending the signal representing the second request.

Advantages that can be seen in implementations of the invention include one or more of the following. A user can delegate authentication tasks to a Web server that requires only a single authentication from the user.

The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 shows a simplified view of an exemplary user screen associated with a corporate portal system, comprising a plurality of content components.

FIG. 2 shows a system for delivering personalized content according to a prior art method.

FIG. 3 shows a system for delivering personalized content according to one implementation.

FIG. 4 shows a process executed by a Web server according to one implementation of the system of FIG. 3.

FIG. 5 shows a system according to an implementation that is integrated with an SSO product.

FIG. 6 shows a process executed by a Web server according to one implementation of the system of FIG. 5.

FIG. 7 shows a system according to an implementation that is integrated with a plurality of SSO products simultaneously.

FIG. 8 shows a process executed by a Web server according to one implementation of the system of FIG. 7.

DETAILED DESCRIPTION

FIG. 3 shows a system 300 for delivering personalized content according to one implementation. A plurality of resource servers 302 and 304 host resource components.

Referring to FIG. 3, a CRM server 302 hosts resources such as customer lists and customer contact information. An email server 304 hosts resources such as email messages for a group of users. A Web server 310 maintains a list of the types of resources available from resource servers 302 and 304, and advertises these types of resources to users. Of course, other types of resources, such as enterprise resource planning resources, can be made available to users.

In one implementation, Web server 310 is a corporate portal. A fundamental purpose of a corporate portal is to allow users a single point of access to all the content, applications, and resources that they need to run their business on a daily basis. This experience is enhanced if users can access all of the resources they are authorized to use through a single act of authentication. Rather than logging on to the portal and then being repeatedly prompted for user name and password as they access various resources, users can move easily between resources, for example shifting with a single click from their email to a query of their CRM application.

Remote users employ user terminals 318A and 318B through 318N to access Web server 310 over a network 320 such as the Internet. A user of a user terminal 318 is referred to as "remote" from Web server 310 because he is not physically operating Web server 310, but rather is operating a user terminal 318 that is physically separate from Web server 310. As used herein, "user terminal" refers to any device that a user could employ to access the Web server including a computer running a Web browser, a personal digital assistant, a cellular phone, and the like.

Web server 310 communicates with each resource server 302 and 304 using a generic application-layer protocol such as HTTP. In one implementation, Web server 310 uses the same protocol for all of the resource servers. Any needed protocol translations are performed at the resource server.

The resource servers can include secure resource servers, such as 304, and open servers, such as CRM server 302. Some secure servers 304 employ a secure front end, referred to herein as an "authenticator," to limit access. To gain access to a secure server, a request must provide a security credential of the type required by the authenticator protecting that secure server. One type of security credential is the userid/password required by HTTP 1.1 basic authentication. When challenged by a web site using HTTP 1.1 basic authentication, a user provides his userid and password. If the userid/password matches the

userid/password stored in an access control list maintained by the server, the user is granted access. The userid/password can be entered manually by the user, or can be passed to the server as a persistent cookie in the HTTP header of a request. After authenticating a user, a secure server may issue the user a session cookie, which the secure server will accept in subsequent requests to authenticate the user for the duration of the user's session with the secure server.

In one implementation, Web server 310 authenticates users according to HTTP 1.1 basic authentication. Web server 310 retains the logon/password for each user, for example in a credentials cache 350. Web server 310 examines each user request to determine which content components are required, and therefore which resource servers must be accessed to provide those content components.

FIG. 4 shows a process 400 executed by Web server 310 according to one implementation of system 300. Process 400 begins when Web server 310 receives a request from a user for a secure resource (step 402). In one implementation, the request is part of a request for personalized content, such as a "my page," that the user has previously defined to contain one or more content components hosted by the secure resource. Web server 310 authenticates the user (step 404). In one implementation, Web server 310 obtains HTTP 1.1 basic authentication security credentials (userid/password) from the request. Web server 310 retains the userid/password for possible subsequent use.

In another implementation, Web server 310 asks the operating system executing on the user terminal 318 for the userid/password of the currently logged-in user, rather than prompting the user. If the user has already logged into her workstation, the operating system sends the login credentials to Web server 310 without user intervention.

When a user logs on to Web server 310, she can choose to have the Web server remember her userid/password so that she does not have to type them in the next time she accesses the Web server. When she logs on to the Web server successfully, the Web server stores her name and password as persistent cookies. These cookies are stored as part of her profile on her computer; whenever she logs on to her computer, the cookie that contains her name and password is made available to her Internet browser. The next time she logs on to her computer and brings up an Internet browser that connects to Web server 310, her browser

sends the cookie with her userid/password to Web server 310, and she is granted access to Web server 310 without having to type in her userid/password.

Based on the request, Web server 310 identifies the resources requested by the user. For each requested resource, Web server 310 determines the type of security credential required for access (step 406). Referring to FIG. 3, Web server 310 determines that CRM server 302 requires no security credentials for access, and that authenticator 324 requires HTTP 1.1 basic authentication security credentials to access email server 304.

Web server 310 maintains a map between each resource server and the type of security credential required to access that resource server, and employs this map to determine the type of security credential required by each server. In one implementation, this map is established by a system administrator when configuring Web server 310. However, this map can be established by other methods. The map can take various forms. For example, a simple true/false flag can be maintained for each resource server. When the flag is true for a resource serve, the userid/logon should be sent. As another example, the path and domain of each secure server requiring an SSO credential (discussed in more detail below) can be stored. Whenever a user request requires access to one of the secure servers, and the path/domain match one of the stored path/domain pairs, the SSO credential for that path/domain are sent. Finally, some secure servers may require custom credentials that neither conform to a standard such as HTTP 1.1, nor to a commercial SSO product. In this case, Web server 310 simply stores the required credentials, which are furnished to the secure server as needed.

In one implementation, Web server 310 stores security credentials for its users, for example in credentials cache 350, or in memory for a single session. Web server 310 can also obtain credentials from other sources, as is well-know in the relevant arts.

When a user's request requires access to a resource on a secure server, Web server 310 retrieves a security credential for the user of the type required to access the secure resource (step 408). Referring to FIG. 3, Web server 310 retrieves the userid/password for the user.

Web server 310 then generates a request for each resource required by the user's request. For requests directed to a secure server, Web server 310 generates a request including the retrieved security credential for the user of the type required by that server (step

410). Web server 310 then sends each request to the appropriate resource server. Because CRM server 302 requires no security credentials for access, a request having no security credentials is sent to CRM server 302, which returns a response including the requested content. Because authenticator 324 requires HTTP 1.1 basic authentication security credentials, a request including the user's userid/password is sent to email server 304 (step 412). In this way, Web server 310 ensures that each resource server receives only those security credentials (if any) required for access. These requests can be sent sequentially or concurrently as described in U.S. Ser. No. 09/949,532 filed September 7, 2001, which is incorporated by reference herein.

After the request sent to email server 304 is authenticated by authenticator 324, email server 304 returns a response including the requested content. Web server 310 receives the responses (step 414), and assembles a result based on the responses. For example, Web server 310 assembles the personalized content requested by the user. Web server 310 sends the result to the user (step 416).

Some implementations can integrate with commercially-available single-sign-on products such as those made by Netegrity, Securant, Oblix and Entrust. These integrations allow a user to log on to the portal without being prompted for security credentials if she has already logged on to another resource protected by the same SSO product. Combined with a variety of login methods, SSO products support intricate authorization rules that permit granular control of what secured resources users can access.

Integration with SSO products has several benefits for administrators of the portal. Portal administrators to choose from various authentication methods—including passwords, biometrics, smart cards and others—that can be mixed and matched for specific objects, thereby providing administrators with highly granular control over access to different applications within the enterprise served by the portal. SSO solutions also offer a centralized way to manage user access and permissions to the portal, as well as other to enterprise applications and content, reducing administrative costs and development time.

In general, all resources secured by an SSO product trust the SSO product to request and process credentials from its users. Most SSO products attach to many kinds of user and group stores, such as LDAP, NT, or ODBC databases. Administrators can choose to protect all or only a few of their Web servers with an SSO product. If the portal has users (such as

business partners) who do not have accounts with the SSO product, these users must log in through a virtual directory that is not protected by SSO. However, if all of the portal users are SSO users, then the administrator can enable SSO on every portal server.

FIG. 5 shows a system 500 according to an implementation that is integrated with an SSO product. A plurality of resource servers 302, 304 and 306 host different types of content components. For example, a stock quotes server 506 hosts content such as stock quotes. Users employ user terminals 318A and 318B through 318N to access Web server 310 over a network 320 such as the Internet as described above. Web server 310 communicates with each resource server using a generic application-layer protocol such as HTTP. In one implementation, Web server 310 uses the same protocol for all of the resource servers. Any needed protocol translations are performed at the resource server.

The resource servers can include secure and open servers. Referring to FIG. 5, a pair of SSO products 532A and 532B protect secure stock quotes server 506 and Web server 310. Another SSO product 532B protects Web server 310. SSO products usually have at least three components: an interceptor 540, an authentication server 542, and a policy server 544. For simplicity, these components are shown only for SSO product 532B.

FIG. 6 shows a process 600 executed by Web server 310 according to one implementation of system 500. Process 600 begins when a user sends a request for content from a secure resource (step 602). An SSO interceptor 540 intercepts the request before it reaches Web server 310 (step 604). Interceptors are usually installed on each server that hosts a secured resource. This secured resource can be a Web page or Web service on a Web site. In a Windows environment, the interceptor can be an ISAPI filter that is installed on the Web site. An ISAPI filter alters each incoming HTTP request before passing it along to the protected Web site to ensure the user is authenticated and authorized. For corporate portal deployments in the UNIX environment, NSAPI or Apache Modules provide analogous functionality.

For instance, an interceptor might be installed on an enterprise's partner Web site, email server, and portal Web servers. Whenever a user attempts to access one of these systems, the interceptor intercepts the request. The interceptor checks whether the user has already been authenticated to at least one of the protected applications. The SSO product checks in a cache of currently logged-in users to see whether the user has logged in recently.

If the user is found, then she is granted access to the secured resource without being asked to enter any additional credentials. If the user does not have a current logged-in session, she will be prompted for credentials.

SSO product 532B authenticates the user, if she hasn't been authenticated already, using an authentication server 542 (step 606). Once a user is authenticated, the SSO product determines whether she is authorized to access the resource being requested (in this case, Web server 310) using policy server 544 (step 608). The process of authorizing access to a secured resource is similar to processing the access control list of a document: some users can read the document, some can write the document, and some can't view the document at all. Most SSO products have a component called a policy server that stores this information about which users and groups can access which secured resources. Although an SSO product's policy server cannot protect portal-specific objects, such as cards that contain metadata, the policy server can protect secure resource servers accessed through the Web server 310.

Once the interceptor has both authenticated and authorized the user, the interceptor adds information to the user's HTTP request and sends the request to the Web site that the user is attempting to access (in this case, Web server 310). In one implementation, this extra information is in the form of HTTP headers, and includes an encrypted security token (referred to herein as an "SSO credential") as well as additional information about the user, such as her name (step 610). Most SSO products allow administrators to configure the information that is added to the HTTP request; for instance, many SSO products allow administrators to pull a user's attributes from an LDAP directory and send them in the HTTP request.

Web server 310 receives the user's request from SSO product 532B (step 612). In one implementation, the request is part of a request for personalized content, such as a "my page," that the user has previously defined to contain one or more content components that include the secure resource. Web server 310 authenticates the user using the SSO credential (step 614). In one implementation, the request also includes the HTTP 1.1 basic authentication security credentials (userid/password) for the user. Web server 310 retains the userid/password for possible subsequent use.

Based on the request, Web server 310 identifies the resources requested by the user. For each requested resource, Web server 310 determines the type of security credential required for access (step 616). Assume that the user request requires content from all of resource servers 302, 304, and 506. Referring to FIG. 5, Web server 310 determines that CRM server 302 requires no security credentials for access, that authenticator 324 requires HTTP 1.1 basic authentication security credentials to access email server 304, and that SSO product 532A requires an SSO token to access stock quotes server 506. This determination is made in a method similar to that described above with respect to FIG. 3.

Web server 310 stores security credentials for its users, for example in credentials cache 350. When a user's request requires access to a resource on a secure server, Web server 310 retrieves a security credential for the user of the type required to access the secure resource (step 618). Referring to FIG. 5, Web server 310 retrieves the userid/password for the user and the SSO token received from SSO product 532B.

Web server 310 then generates a request for each resource required by the user's request. For requests directed to a secure server, Web server 310 generates a request including the retrieved security credential for the user of the type required by that server (step 620). Web server 310 then sends each request to the appropriate resource server. A request having no security credentials is sent to CRM server 302, which returns a response including the requested content. A request including the proper security credentials is sent to each required secure server (step 622). A request including the user's userid/password is sent to email server 304. A request including the SSO token is sent to stock quotes server 506. In this manner, Web server 310 ensures that each resource server receives only those security credentials (if any) required for access.

Each secure resource server authenticates the request using the supplied security credentials. Each resource server then returns a response including the requested content. Web server 310 receives the responses (step 624), and assembles a result based on the responses. For example, Web server 310 assembles the personalized content requested by the user. Web server 310 sends the result to the user (step 626).

Some implementations can integrate with a plurality of SSO products simultaneously. FIG. 7 shows a system 700 according to such an implementation. A plurality of resource servers 302, 304 and 506 host different types of content components, as described above. In

addition, a web search server 708 provides services such as web search capabilities. Users employ user terminals 318A and 318B through 318N to access Web server 310 over a network 320 such as the Internet, as described above. Web server 310 communicates with each resource server using a generic application-layer protocol such as HTTP. In one implementation, Web server 310 uses the same protocol for all of the resource servers. Any needed protocol translations are performed at the resource server.

The resource servers can include secure and open servers. Referring to FIG. 7, a pair of SSO products 732A and 732B protect secure stock quotes server 506 and Web server 310. SSO products 732 include an interceptor 740A, an authentication server 742A, and a policy server 744A. For simplicity, these components are shown only for SSO product 732B. In addition, a pair of different SSO products 734A and 734B protect secure Web search server 708 and Web server 310. SSO products 734 include an interceptor 740B, an authentication server 742B, and a policy server 744B. For simplicity, these components are shown only for SSO product 734B. Note that SSO products 732 and 734 are different SSO products, for example, manufactured by different SSO vendors.

FIG. 8 shows a process 800 executed by Web server 310 according to one implementation of system 700. Process 800 begins when a user sends a request for content hosted by a secure resource (step 802). The first SSO interceptor 740A intercepts the request before it reaches Web server 310 (step 804). SSO product 732B authenticates the user, if she hasn't been authenticated already, using authentication server 742A (step 806). Once a user is authenticated, the SSO product determines whether she is authorized to access the resource being requested (in this case, Web server 310) using policy server 744A (step 808).

Once the interceptor has both authenticated and authorized the user, the interceptor adds information to the user's HTTP request and sends the request to the Web site that the user is attempting to access (in this case, Web server 310). In one implementation, this extra information is in the form of HTTP headers, and includes an encrypted security token (referred to herein as an "SSO credential") as well as additional information about the user, such as her name (step 810).

The second SSO interceptor 740B intercepts the request before it reaches Web server 310 (step 812) and authenticates the user (step 814). If SSO product 734 has been configured to trust SSO product 732, then SSO product 734B can authenticate the user based on the SSO

credential included in the request by SSO product 732B. If not, then SSO product 734B can authenticate the user based on the user's login credentials, or a session cookie based on those credentials. If necessary, SSO product 734B can authorize the user as well (step 816).

Once interceptor 704B has authenticated the user, interceptor 704B adds a SSO credential to the user's HTTP request in a manner similar to that described above, and sends the request to Web server 310 (step 818).

Web server 310 receives the user's request from SSO product 734B (step 820). In one implementation, the request is part of a request for personalized content, such as a "my page," that the user has previously defined to contain one or more content components that include the secure resource. Web server 310 authenticates the user using the SSO credential generated by SSO product 732B (step 822). In one implementation, the request also includes the HTTP 1.1 basic authentication security credentials (userid/password) for the user. Web server 310 retains the userid/password for possible subsequent use.

Based on the request, Web server 310 identifies the resources requested by the user. For each requested resource, Web server 310 determines the type of security credential required for access (step 824). Assume that the user request requires content from all of resource servers 302, 304, 506 and 708. Referring to FIG. 7, Web server 310 determines that CRM server 302 requires no security credentials for access, that authenticator 324 requires HTTP 1.1 basic authentication security credentials to access email server 304, and that SSO products 732A and 734A each require a SSO token. This determination is made in a method similar to that described above with respect to FIG. 5.

Web server 310 stores security credentials for its users, for example in credentials cache 350. When a user's request requires access to a resource on a secure server, Web server 310 retrieves a security credential for the user of the type required to access the secure resource (step 826). Referring to FIG. 7, Web server 310 retrieves the userid/password for the user and the SSO tokens received from SSO products 732B and 734B.

Web server 310 then generates a request for each resource required by the user's request. For requests directed to a secure server, Web server 310 generates a request including the retrieved security credential for the user of the type required by that server (step 828). Web server 310 then sends each request to the appropriate resource server. A request having no security credentials is sent to CRM server 302, which returns a response including

the requested content. A request including the proper security credentials is sent to each required secure server (step 830). A request including the user's userid/password is sent to email server 304. A request including the SSO token generated by SSO product 732B is sent to stock quotes server 506. A request including the SSO token generated by SSO product 734B is sent to Web search server 708. In this manner, Web server 310 ensures that each resource server receives only those security credentials (if any) required for access.

Each secure resource server authenticates the request using the supplied security credentials. Each resource server then returns a response including the requested content. Web server 310 receives the responses (step 832), and assembles a result based on the responses. For example, Web server 310 assembles the personalized content requested by the user. Web server 310 sends the result to the user (step 834).

The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps of the invention can be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on input data and generating output. The invention can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program can be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language can be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Generally, a computer will include one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices;

magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits).

A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention.

For example, various trust relationships can be established between SSO products such that one type of SSO product will accept tokens generated by another type of SSO product, as is well-known in the relevant art. In such cases, it is not necessary to add every type of token to each user request. For example, if SSO product A is configured to accept the tokens of SSO product B, then it is not necessary to provide an interceptor for SSO product A. Accordingly, other embodiments are within the scope of the following claims.